I hereby certify that this correspondence is being filed via EFS-Web with the United States Patent and Trademark Office on December 4, 2008

TOWNSEND and TOWNSEND and CREW LLP

By: /Kristin M. Brennan/

PATENT
Attorney Docket No. 15114H-071400US
Client Ref. No. A100052GB00/DCO

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	Confirmation No. 1395
DHANOA, Kulwinder	Examiner: Chun Kuan Lee
Application No.: 10/749,910	Technology Center/Art Unit: 2181
Filed: December 30, 2003	APPELLANTS' BRIEF UNDER 37 CFR §41.37
For: SDRAM CONTROLLER	

Mail Stop Appeal Brief
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Further to the Notice of Panel Decision from Pre-Appeal Brief Review mailed on November 4, 2008 for the above-referenced application, Appellants submit this Brief on Appeal pursuant to 37 C.F.R. §41.37. The Commissioner is authorized to deduct from deposit account number 20-1430 any additional fees associated with this brief.

TABLE OF CONTENTS

1. REAL PARTY IN INTEREST

All right, title, and interest in the corresponding application is assigned to Altera Corporation. The real party in interest of the subject patent application thus is Altera Corporation.

2. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences.

3. STATUS OF CLAIMS

Claims 1, 2, 5-8, and 11-13 are pending. Claims 3, 4, 9, 10, and 14-17 are canceled. Appellants appeal from the rejection of claims 1, 2, 5-8, and 11-13.

4. STATUS OF AMENDMENTS

Appellants have not submitted any amendments subsequent to the Final Office Action mailed May 21, 2008.

5. SUMMARY OF CLAIMED SUBJECT MATTER

The following is a concise explanation of the subject matter defined in each of the independent claims, and claim 2, involved in the present Appeal. Where deemed appropriate, the explanation includes references to the specification by page and line number, as well as references to the appropriate drawing(s) and reference character(s). The references are meant to be examples only, and do not include references to every recitation or suggestion of such elements in the specification and/or drawings. The use of these example references is not intended to limit specific claims to specific embodiments or descriptions contained therein.

Claim 1

Claim 1 is directed to a memory controller (See, *e.g.*, Specification, page 1, lines 5-10 and Figure 2). The memory controller has at least one bus interface, with each bus interface being for connection to at least one respective device for receiving memory access requests (See,

e.g., Specification, page 5, lines 1-4 and Figure 2). The memory controller also has a memory interface, for connection to a memory device over a memory bus (See, *e.g.*, Specification, page 5, lines 24-31 and Figure 2). There are a plurality of buffers in the memory interface, with each of the plurality of buffers sized to store a data burst for a memory access request and each of the plurality of buffers further including a plurality of sub-buffers each sized to store a data beat of the data burst stored in the corresponding buffer (See, *e.g.*, Specification, page 1, lines 5-10, page 9, lines 1-6, and Figure 2). The memory controller further includes control logic, for placing received memory access requests into a queue of memory access requests (See, *e.g.*, Specification, page 5, lines 6-9 and lines 17-22, and Figure 2). In response to a received memory access request requiring multiple data bursts over the memory bus, each of said multiple data bursts is assigned by the control logic to a respective buffer of the plurality of buffers in the memory interface, and data from each of said multiple data bursts is stored by the memory interface in the respective buffer (See, *e.g.*, Specification, page 7, lines 9-11 and Figure 3). For a wrapping memory access request requiring multiple buffers, data required for each of a beginning and an end of the wrapping memory access request are assigned to respective sub-buffers of a single respective buffer by the control logic, the beginning and end data for the memory access request being stored concurrently from a single data burst in the respective sub-buffers of the single respective buffer by the memory interface, the storing of the beginning and end data in a single buffer avoiding the need for an additional data burst to obtain the end data, the data required for the end of the wrapping memory access request being cached in the respective sub-buffer until needed for transfer in response to the wrapping memory access request (See, *e.g.*, Specification, page 8, lines 30-36, page 9, lines 1-12, and Figure 4). The control logic records a value of a pointer indicating a first sub-buffer of the single buffer storing the end data, such that the control logic is able to return to the indicated sub-buffer to retrieve the end data from the single respective buffer (See, *e.g.*, Specification, page 9, lines 24-27 and Figure 4).

Claim 2

Claim 2 depends on claim 1 and additionally recites "wherein, when returning data to the respective device from which a memory access request requiring multiple data bursts over the memory bus was received, data is read out from a first part of the single buffer, then data is read out from at least one other of said buffers, then data is read out from a second part of the single buffer." (See, *e.g.*, Specification, page 9, lines 14-36 and page 10, lines 1-6).

Claim 7

Claim 7 is directed to a method of using a memory controller (See, *e.g.*, Specification, page 1, lines 5-10 and Figure 2). The memory controller has at least one bus interface for connection to at least one respective device for receiving memory access requests (See, *e.g.*, Specification, page 5, lines 1-4 and Figure 2). The memory controller also has a memory interface for connection to a memory device over a memory bus (See, *e.g.*, Specification, page 5, lines 24-31 and Figure 2). There is a plurality of buffers in the memory interface, and the memory controller has control logic for placing received memory access requests into a queue of memory access requests (See, *e.g.*, Specification, page 5, lines 6-9 and lines 17-22, and Figure 2). In response to a received memory access request requiring multiple data bursts over the memory bus, the method assigns each data burst to a respective buffer in a plurality of buffers in the memory interface (See, *e.g.*, Specification, page 7, lines 9-11 and Figure 3). Each of the plurality of buffers is sized to store a data burst for the memory access request, and each of the plurality of buffers further includes a plurality of sub-buffers each sized to store a data beat of the data burst stored in the corresponding buffer (See, *e.g.*, Specification, page 8, lines 30-36, page 9, lines 1-6, and Figure 4). The method further includes storing data from each of said multiple data bursts in the respective buffer in the memory interface (See, *e.g.*, Specification, page 7, lines 9-11 and Figure 3). For a wrapping memory access request, the method assigns data required for a beginning and an end of the wrapping memory access request to respective sub-buffers of a single respective buffer to be stored concurrently from a single data burst in the respective sub-buffers of the single respective buffer in the memory interface, the storing of the beginning and end data in a single buffer avoiding the need for an additional data

burst to obtain the end data (See, *e.g.*, Specification, page 8, lines 30-36, page 9, lines 1-12, and Figure 4). The data required for the end of the wrapping memory access request is cached in the respective sub-buffer until needed for transfer in response to the wrapping memory access request (See, *e.g.*, Specification, page 10, lines 80-12, and Figure 4). The method further provides for recording a value of a pointer indicating a first sub-buffer of the single buffer storing the end data, and using the pointer to return to the indicated sub-buffer to retrieve the end data.(See, *e.g.*, Specification, page 9, lines 24-27 and Figure 4).

Claim 13

Claim 13 is directed to a programmable logic device that has a memory controller (See, *e.g.*, Specification, page 1, lines 5-10 and Figure 2). The programmable logic device has at least one bus interface, with each bus interface being for connection to at least one respective device formed within the programmable logic device for receiving memory access requests (See, *e.g.*, Specification, page 5, lines 1-4 and Figure 2). The programmable logic controller also has a memory interface, for connection to an external memory device over a memory bus (See, *e.g.*, Specification, page 5, lines 24-31 and Figure 2). There are a plurality of buffers in the memory interface, with each of the plurality of buffers sized to store a data burst for a memory access request (See, *e.g.*, Specification, page 1, lines 5-10 and Figure 2). Each of the plurality of buffers further include a plurality of sub-buffers each sized to store a data beat of the data burst stored in the corresponding buffer (See, *e.g.*, Specification, page 9, lines 1-12, and Figure 2). The programmable logic device further includes control logic, for placing received memory access requests into a queue of memory access requests (See, *e.g.*, Specification, page 5, lines 6-9 and lines 17-22, and Figure 2). In response to a received memory access request requiring multiple data bursts over the memory bus, each of said multiple data bursts is assigned by the control logic to a respective buffer of the plurality of buffers in the memory interface, and data from each of said multiple data bursts is stored by the memory interface in the respective buffer (See, *e.g.*, Specification, page 7, lines 9-11 and Figure 3). For a wrapping memory access request requiring multiple buffers, data required for each of a beginning and an end of the wrapping memory access request are assigned to respective sub-buffers of a single respective buffer by the

control logic, with the beginning and end data for the memory access request being stored concurrently from a single data burst in the respective sub-buffers by the memory interface (See, *e.g.*, Specification, page 8, lines 30-36, page 9, lines 1-12, and Figure 4). The storing of the beginning and end data in a single buffer avoids the need for an additional data burst to obtain the end data, and the data required for the end of the wrapping memory request is cached in the respective sub-buffer until needed for transfer in response to the wrapping memory access request (See, *e.g.*, Specification, page 8, lines 30-36, page 9, lines 1-12, and Figure 4). The control logic records a value of a pointer indicating a first sub-buffer of the single buffer storing the end data, such that the control logic is able to return to the indicated sub-buffer to retrieve the end data from the single buffer (See, *e.g.*, Specification, page 9, lines 24-27 and Figure 4).

6. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Issue: Whether claims 1, 2, 7-8, and 13 were properly rejected under 35 U.S.C. 103(a) as being obvious over *Gray* (U.S. Pat. No. 6,816,923) in view of *Iizuka* (U.S. Pat. No. 5,581,530) and further in view of *Nguyen* (U.S. Pat. No. 5,335,326).

Also, whether claims 5 and 11 were properly rejected under 35 U.S.C. § 103(a) as being unpatentable over *Gray*, in view of *Iizuka*, in view of *Nguyen*, and further in view of *Kuronuma* (U.S. Patent No. 6,859,848).

Also, whether claims 6 and 12 were properly rejected under 35 U.S.C. § 103(a) as being unpatentable over *Gray*, in view of *Iizuka*, in view of *Nguyen*, and further in view of *Microsoft Computer Dictionary*.

For the purposes of this appeal, *Gray, Iizuka,* and *Nguyen* will be treated as prior art. However, Appellants reserve the right to later disqualify these references as prior art. Furthermore, for purposes of this appeal, claims 5 and 6 may stand or fall with claim 1, and claims 8, 11, and 12 may stand or fall with claim 7. Appellants would like to separately argue the patentability of claim 2. No admissions are made by the groupings of claims, and Appellants reserve the right to pursue features in any of the claims in one or more continuation applications.

7. ARGUMENT

Rejection under 35 U.S.C. § 103(a) as being obvious over Gray, Iizuka, and Nguyen.

Overview of Appellants' Technique Pertaining to the Pending Claims

Conventional computer systems must be provided with memory devices containing sufficient data storage capacity to operate correctly. The accessing of a memory device is performed by a memory controller. The memory controller is connected to the memory device by means of a memory data bus, and a goal of the memory controller is to make efficient use of the bandwidth of the memory bus for rapid memory access. (See, *e.g.*, Specification, page 1, lines 13-25). Access requests received by a memory controller will specify the amount of data to be retrieved from the memory device. Data is received from the memory device in bursts, with each burst containing a fixed amount of data, and occupying the memory bus for a corresponding fixed time period. In the case of a request to read data from the memory device, the access request will also specify whether it is a wrapping read request. In a wrapping read request, the data to be read from the memory device is stored at memory locations in the memory device, with the addresses of those memory locations returning to near the start point towards the end of the read operation. (See, *e.g.*, Specification, page 2, lines 1-8). In a conventional system, this has the consequence that only a part of the data returned from the memory device in the first data burst is passed to the requesting device, and that the same data burst is requested again at the end of the read operation to allow the remaining data to be passed to the requesting device. This results in inefficient usage of the available bandwidth of the memory bus. (See, *e.g.*, Specification, page 2, lines 8-16).

The pending claims relate to memory controllers and methods using memory controllers. In one embodiment, when a single memory access request requires multiple data bursts on the memory bus, the memory controller stores the data from the multiple data bursts in multiple buffers (See, *e.g.*, Specification, page 5, lines 24-31). Data for the first part of the request and the last part of the request can be written concurrently to a common first buffer on a single pass, such that no additional pass is needed as in conventional systems to separately store

the data for the end of the request. ("assigning data required for a beginning and an end of the wrapping memory access request to respective sub-buffers of a single respective buffer to be stored concurrently from a single data burst in the respective sub-buffers of the single respective buffer in the memory interface, the storing of the beginning and end data in a single buffer avoiding the need for an additional data burst to obtain the end data" as recited in claim 7). Data is then retrieved from the buffers such that data is read from a part of the first buffer, then from the other buffers, and finally from the remaining part of the first buffer. (See, *e.g.,* Specification, page 9, lines 8-22 and FIG. 4). Storing the required data in the remaining part of the first buffer as discussed above thus avoids the need to occupy the memory bus with a new data burst. A pointer is used to keep track of the part of the first buffer that the data is initially read from, to allow for the final reading of the end of the request. (See, *e.g.,* Specification, page 9, lines 24-27 and FIG. 4). This has advantages, including that the overall performance of the computer system is improved since a higher bandwidth can be achieved on the memory data bus, thereby allowing the memory to be used more efficiently.

Overview of the Cited Prior art References

Gray teaches a direct memory access (DMA) system including a DMA engine that includes a data reservoir having a number of memory buffers in order to consolidate memory buffers. The single reservoir includes portions that correspond to different devices (col. 2, lines 34-47; col. 3, line 64-col. 4, line 18). The use of the consolidated memory reservoir also provides the ability to centralize addressing and provide each device with data in a timely manner and with increased bandwidth (col. 2, lines 34-56). *Gray* consolidates memory into buffers in the data reservoir and does not utilize buffers in a memory interface, instead teaching replacing buffers for individual devices with the reservoir (shown in Fig. 3; also col. 2, lines 34-47; col. 4, lines 10-12). *Gray* includes a plurality of different device-specific buffers in the DMA engine, so that requests for separate devices are sent to separate buffers (Fig. 3; col. 8, lines 10-21). *Gray* does not teach or suggest different buffers for different portions of a single request from a single device as recited in Appellants' claim 1. Further, as recognized in the final Office Action on page 4, *Gray* does not teach or suggest a request requiring multiple data bursts

and a wrapping memory access request requiring multiple buffers, with data required for a beginning and an end of the request being stored in a single buffer, and a pointer.

Nguyen is directed to a bus to bus interface. More specifically, *Nguyen* discloses a bus-to-bus interface including a central buffer means including first and second FIFO devices, utilizing a pointer means to traverse a circular queue in the FIFO devices slot by slot. (Nguyen, col. 1, line 59 - col. 2, line 49.)

Iizuka is directed to a digital recorder for processing parallel data stored in multiple tracks. *Iizuka* teaches a hard disk buffer, and different tracks on the hard drive can be temporarily stored in individual buffers to increase hard disk access speed. (*Iizuka*, abstract and col. 19, lines 56-60).

Claims 1, 5-8, and 11-13

The pending rejection does not establish prima facie obviousness under 35 U.S.C. § 103 and M.P.E.P. §§ 2142-2143. The Examiner argues on page 5 of the final Office Action dated May 21, 2008 (the "final Office Action") that

> [i]t would have been obvious to one of ordinary skill in this art, at the time of invention was made to include Iizuka's buffering architecture into Gray's device buffers for the benefit of implementing a simplified structure and providing an optimal priority order for data transferring (Iizuka. col. 2, II. 61-67) to obtain the invention as specified in claims 1, 7 and 13.

Thus, the Examiner appears to apply the rationale for obviousness rejections of "Combining Prior Art Elements According to Known Methods to Yield Predictable Results." See, e.g., M.P.E.P. §2131. M.P.E.P. 2143A provides, for such a rationale, that

> [o]ffice personnel must articulate the following:
> (1) a finding that the prior art included each element claimed, although not necessarily in a single prior art reference, with the only difference between the claimed invention and the prior art being the lack of actual combination of the elements in a single prior art reference;
> (2) a finding that one of ordinary skill in the art could have combined the elements as claimed by known methods, and that in

> combination, each element merely performs the same function as it does separately;
> (3) a finding that one of ordinary skill in the art would have recognized that the results of the combination were predictable; and
> (4) whatever additional findings based on the Graham factual inquiries may be necessary, in view of the facts of the case under consideration, to explain a conclusion of obviousness.

Appellants respectfully submit that a prima facie case of obviousness has not been met because the Examiner's rejection fails on at least one of the above requirements, as explained in more detail below.

A. *plurality of buffers for a memory access request*

Claim 1 recites,

> a plurality of buffers in the memory interface, **each of the plurality of buffers sized to store a data burst for a memory access request, each of the plurality of buffers further including a plurality of sub-buffers** each sized to store a data beat of the data burst stored in the corresponding buffer;
> and
> each of said multiple data bursts is assigned by the control logic to a respective buffer of the plurality of buffers in the memory interface, and data from each of said multiple data bursts is stored by the memory interface in the respective buffer (*emphasis added*).

The Office Action admits that *Gray* does not teach a plurality of buffers sized to store a data burst for a single memory access request, and is silent as to *Nguyen*. Appellants submit that *Nguyen* also does not teach a plurality of buffers each sized to store a data burst for a memory access request, particularly where each buffer further includes a plurality of sub-buffers each sized to store a data beat of the data burst stored in the corresponding buffer (See page 4 of the final Office Action).

On page 4, the Final Office Action cites *Iizuka* as teaching these features. *Iizuka* discloses a buffer with respect to a hard disk storage device to compensate for slower data transfer between the hard disk and an I/O device. (*Iizuka*, Abstract.) For example, in FIGS. 8 and 14, *Iizuka* discloses three independent audio tracks of data (tr1, tr2, and tr3) that interface with a hard disk, and each independent track is associated with a respective buffer (buffers 9-1 to

9-3). (*Iizuka*, col. 11, lines 21-26.) *Iizuka* discloses that each buffer is independent of other respective buffers because each track of data is independent from other tracks. Thus, each independent I/O device 8-1 to 8-3 will separately exchange a respective one of tr1, tr2, and tr3 with a respective buffer 9-1 to 9-3. See *Iizuka*, col. 10, lines 46-51 and 64-67.

In other words, track 1 may be receiving data from the hard disk while tracks 2 and 3 may be transmitting data to the hard disk. (*Iizuka*, col. 14, lines 36-40.) To allow for this independent operation, *Iizuka* teaches that each memory access request between a track and the hard disk is associated with the individual track, not a combination of tracks. (*Iizuka*, col. 14, lines 29-40.) Furthermore, because each track of data is independent from the other tracks, each respective buffer of each track is also independent. (*Iizuka*, col. 10, lines 46-51.) Thus, *Iizuka* teaches use of a **single** buffer associated with each single memory access (or DMA transfer).

Iizuka does not show or provide a reason to use a plurality of buffers for a single memory request, and both *Gray* and *Nguyen* are silent on the matter. Therefore, the cited prior art references do not render obvious, alone or in combination, a "plurality of buffers sized to store a data burst for a memory access request, each of the plurality of buffers further including a plurality of sub-buffers each sized to store a data beat of the data burst stored in the corresponding buffer" as recited in claim 1.

B. *wrapping memory access request*

Claim 1 also recites:

> for a **wrapping memory access request** requiring multiple buffers, data required for each of a beginning and an end of the wrapping memory access request are assigned to respective sub-buffers of a single respective buffer by the control logic, **the beginning and end data for the memory access request being stored concurrently from a single data burst** in the respective sub-buffers of the single respective buffer by the memory interface
> (*emphasis added*).

The cited prior art does not teach, suggest, or otherwise render obvious the above limitations. The Examiner admits, on page 4 of the final Office Action with regards to claim 5, that *Gray* does not teach that the wrapping memory access request requires multiple buffers and that data required for each of a beginning and an end of the wrapping memory access request are

assigned to respective sub-buffers of a single respective buffer. Furthermore, Appellants submit that neither *Nguyen* nor *Iizuka* teach a wrapping memory access request as claimed.

On page 4, the Final Office Action cites *Iizuka* as teaching these features. However, at page 7 of the final Office Action, the Examiner admits that "*Gray, Iizuka* and *Nguyen* does not expressly teach the memory controller system, method and programmable logical device, comprising wherein the control logic determines whether a received read access request is a wrapping request which requires multiple memory bursts." For at least this reason, independent claims 1, 7, and 13 should be allowable. Furthermore, as argued above, *Iizuka* discloses only a single buffer associated with each memory access. As such, the Office Action assertion on page 4 that a DMA transfer is equivalent to a "wrapping memory access request" fails because claim 1 recites "a wrapping memory access request requiring multiple buffers." As discussed above, a DMA transfer moves data to/from a single buffer. (*Iizuka*, col. 26, lines 4-14.) Because a DMA transfer discloses a single buffer, a DMA transfer cannot teach a wrapping memory access request requiring multiple buffers, as recited in claim 1.

Furthermore, *Iizuka* does not teach or suggest "data required for each of a beginning and an end of the wrapping memory access request are assigned to respective sub-buffers of a single respective buffer." On page 5, the final Office Action asserts that the data block shown in FIG. 8 discloses beginning and end data in a single buffer 9-1, but this assertion only applies if the entire memory access request in FIG. 8 is contained in a single buffer. See column 14, lines 50-57. As discussed above, claim 1 recites a wrapping memory access request requiring multiple buffers (i.e., portions of the wrapping memory access request are not in the same buffer as the beginning and end portions). The individual buffers as shown in FIG. 8 of *Iizuka* do not disclose both beginning and end data in a single buffer for a wrapping memory access request, where that single request requires multiple buffers.

Even assuming, arguendo, that the buffers associated with the other audio tracks are considered to comprise "multiple buffers" that receive a single memory access request, *Iizuka* still does not teach beginning and end data for multiple audio tracks being assigned to a single respective buffer. As shown in *Iizuka* FIGS. 14(a)-(e), when sending data tracks to a hard disk, the data is transferred from the first buffer in FIG.14(a). Then, data is transferred from the

second buffer in FIG.14(b). Finally, data is transferred from a third buffer in FIG.14(c). See column 26, lines 4-24. As seen in FIGS. 14(a)-(e), the beginning data is transferred from a first buffer 9-1, and end data is transferred from a third buffer 9-3. There is no teaching or suggestion for beginning and end data assigned to a single buffer as recited in claim 1.

Iizuka does not also render obvious the limitation of claim 1 regarding "the beginning and end data for the memory access request being stored **concurrently from a single data burst** in the respective sub-buffers of the single respective buffer by the memory interface." (emphasis added). As shown in FIGS. 14(a)-(e) and column 26, lines 4-24, *Iizuka* discloses that data is transferred from the buffers in a sequential manner starting with track 1, then track 2, then track 3, and then back to track 1, etc. Because of the sequential nature of the data transfer, *Iizuka* discloses that the beginning data for the first track would be transferred at a first time, and the end data would be transferred at a subsequent time. Therefore, *Iizuka* cannot teach a **concurrent** data transfer of beginning and end data of a memory access request requiring multiple buffers.

C. *pointer indicating the end data*

The prior art also fails to show a "a pointer indicating a first sub-buffer of the single buffer storing the end data" as recited in claim 1. The Examiner cites *Nguyen* on pages 5-6 of the final Office Action for teaching a pointer "for pointing to the proper slot for the next input operation."

However, *Nguyen* fails to provide such teaching to make up for the deficiencies in the references with respect to Appellant's claim 1. The Office Action combines *Nguyen* in an attempt to show the use of a pointer as recited in Appellant's claim 1. *Nguyen* teaches a bus-to-bus interface including a central buffer means including first and second FIFO devices, utilizing a pointer means to traverse a circular queue in the FIFO devices slot by slot. (*Nguyen*, col. 1, line 59 - col. 2, line 49.) Even if for sake of argument it would have been obvious to combine the pointer of Nguyen with the combination of *Gray* and *Iizuka* set forth above, the pointer (*Nguyen*) would still at best keep track of the next sub-buffer in the single circular buffer (*Iizuka*) for each device (*Gray*).

The recited references cannot in any way be combined to show that "the beginning and end data for the memory access request being stored concurrently from a single data burst in the respective sub-buffers of the single respective buffer...wherein the control logic records a value of a pointer indicating a first sub-buffer of the single buffer storing the end data," as recited in claim 1. .

Thus, even if for sake of argument there is a rationale to combine *Iizuka* with *Gray* and *Nguyen*, the combination would still fail to teach, suggest, or otherwise render obvious each and every element of claim 1. For at least these reasons, the rejection over claim 1 should be withdrawn. As claim 1 is allowable, dependent claims 2, 5, and 6 are also allowable for at least the same rationale.

Appellant submits that independent claims 7 and 13 should be allowable for at least some of the reasons mentioned above with respect to claim 1. For example, Appellants' claim 7 recites, in a memory controller including at least one bus interface for connection to at least one respective device for receiving memory access requests, a memory interface for connection to a memory device over a memory bus, a plurality of buffers in the memory interface, and control logic for placing received memory access requests into a queue of memory access requests, a method of retrieving data comprising:

> in response to a received memory access request requiring multiple data bursts over the memory bus, **assigning each data burst to a respective buffer** in a plurality of buffers in the memory interface, **each of the plurality of buffers being sized to store a data burst for the memory access request, each of the plurality of buffers further including a plurality of sub-buffers each sized to store a data beat of the data burst stored in the corresponding buffer**;
> storing data from each of said multiple data bursts in the respective buffer in the memory interface;
> **for a wrapping memory access request, assigning data required for a beginning and an end of the wrapping memory access request to respective sub-buffers of a single respective buffer to be stored concurrently from a single data burst in the respective sub-buffers of the single respective buffer in the memory interface, the storing of the beginning and end data in a single buffer avoiding the need for an additional data burst**

> **to obtain the end data, the data required for the end of the wrapping memory access request being cached in the respective sub-buffer until needed for transfer in response to the wrapping memory access request;**
> **recording a value of a pointer indicating a first sub-buffer of the single buffer storing the end data**; and
> **using the pointer to return to the indicated sub-buffer to retrieve the end data**
> (*emphasis added*).

For reasons including at least some of those discussed above with respect to claim 1, the references do not teach, suggest, or otherwise render obvious these elements of Appellants' claim 7.

Similarly, Appellants' claim 13 recites a programmable logic device, wherein the programmable logic device includes a memory controller, comprising:

> at least one bus interface, each bus interface being for connection to at least one respective device formed within the programmable logic device for receiving memory access requests;
> a memory interface, for connection to an external memory device over a memory bus;
> **a plurality of buffers in the memory interface, each of the plurality of buffers sized to store a data burst for a memory access request, each of the plurality of buffers further including a plurality of sub-buffers each sized to store a data beat of the data burst stored in the corresponding buffer**; and
> control logic, for placing received memory access requests into a queue of memory access requests,
> wherein, **in response to a received memory access request requiring multiple data bursts over the memory bus, each of said multiple data bursts is assigned by the control logic to a respective buffer of the plurality of buffers in the memory interface, and data from each of said multiple data bursts is stored by the memory interface in the respective buffer**, and
> wherein, **for a wrapping memory access request requiring multiple buffers, data required for each of a beginning and an end of the wrapping memory access request are assigned to respective sub-buffers of a single respective buffer by the control logic, the beginning and end data for the memory access request being stored concurrently from a single data**

> **burst in the respective sub-buffers** by the memory interface, **the storing of the beginning and end data in a single buffer avoiding the need for an additional data burst to obtain the end data, the data required for the end of the wrapping memory request being cached in the respective sub-buffer until needed for transfer in response to the wrapping memory access request**; and
> wherein **the control logic records a value of a pointer indicating a first sub-buffer of the single buffer storing the end data, such that the control logic is able to return to the indicated sub-buffer to retrieve the end data from the single buffer.**
> (*emphasis added*).

For reasons including at least some of those discussed above with respect to claim 1, the references do not teach, suggest, or otherwise render obvious these elements of Appellants' claim 13.

As claim 7 is allowable, dependent claims 8, 11, and 12 are allowable for at least the same rationale.

Claim 2

Dependent claim 2 is also clearly allowable over the prior art. At page 6 of the final Office Action, the Examiner argues that *Gray*, in FIG. 3 and column 12, lines 18-30, shows "data is read out from a first part of the single buffer, then data is read out from at least one other of said buffers, then data is read out from a second part of the single buffer" as recited in claim 2. However, *Gray* only discloses requesting data from buffers at different cycles, to allow for scheduling various devices to have access to a limited number of buffers. See column 12, lines 31-44. These buffer reads are for the entire buffer, not portions. *Gray* does not teach or suggest first reading data from a *first* part of a buffer, then reading data from a *second* part of the same buffer as claimed. Furthermore, Appellants' submit that both *Iizuka* and *Nguyen* are silent as to the recited features. Thus, the references do not teach, suggest, or otherwise render obvious claim 2.

8. CONCLUSION

For all the foregoing reasons, it is respectfully requested that the Board of Patent Appeals and Interferences reverse the rejections of the Examiner regarding claims 1, 2, 5-8, and 11-13 so that this case may be allowed and pass to issue in a timely manner.

Respectfully submitted,

Adam J. Pyonin
Reg. No. 57,301

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, Eighth Floor
San Francisco, California 94111-3834
Tel: 650-326-2400
Fax: 650-326-2422
61700436 v4

9. CLAIMS APPENDIX

1. A memory controller, comprising:

at least one bus interface, each bus interface being for connection to at least one respective device for receiving memory access requests;

a memory interface, for connection to a memory device over a memory bus;

a plurality of buffers in the memory interface, each of the plurality of buffers sized to store a data burst for a memory access request, each of the plurality of buffers further including a plurality of sub-buffers each sized to store a data beat of the data burst stored in the corresponding buffer; and

control logic, for placing received memory access requests into a queue of memory access requests,

wherein, in response to a received memory access request requiring multiple data bursts over the memory bus, each of said multiple data bursts is assigned by the control logic to a respective buffer of the plurality of buffers in the memory interface, and data from each of said multiple data bursts is stored by the memory interface in the respective buffer, and

wherein, for a wrapping memory access request requiring multiple buffers, data required for each of a beginning and an end of the wrapping memory access request are assigned to respective sub-buffers of a single respective buffer by the control logic, the beginning and end data for the memory access request being stored concurrently from a single data burst in the respective sub-buffers of the single respective buffer by the memory interface, the storing of the beginning and end data in a single buffer avoiding the need for an additional data burst to obtain

the end data, the data required for the end of the wrapping memory access request being cached in the respective sub-buffer until needed for transfer in response to the wrapping memory access request, and

wherein the control logic records a value of a pointer indicating a first sub-buffer of the single buffer storing the end data, such that the control logic is able to return to the indicated sub-buffer to retrieve the end data from the single respective buffer.

2. A memory controller as claimed in claim 1, wherein, when returning data to the respective device from which a memory access request requiring multiple data bursts over the memory bus was received, data is read out from a first part of the single buffer, then data is read out from at least one other of said buffers, then data is read out from a second part of the single buffer.

Claims 3-4. (Canceled)

5. A memory controller as claimed in claim 1, wherein the control logic determines whether a received read access request is a wrapping request which requires multiple memory bursts, and, if so, the control logic allocates each of said memory bursts to a respective one of said buffers.

6. A memory controller as claimed in claim 1, wherein the memory controller is a SDRAM controller, and said memory interface is suitable for connection to a SDRAM memory device over said memory bus.

7. In a memory controller including at least one bus interface for connection to at least one respective device for receiving memory access requests, a memory interface for connection to a memory device over a memory bus, a plurality of buffers in the memory interface, and control logic for placing received memory access requests into a queue of memory access requests, a method of retrieving data comprising:

in response to a received memory access request requiring multiple data bursts over the memory bus, assigning each data burst to a respective buffer in a plurality of buffers in the memory interface, each of the plurality of buffers being sized to store a data burst for the memory access request, each of the plurality of buffers further including a plurality of sub-buffers each sized to store a data beat of the data burst stored in the corresponding buffer;

storing data from each of said multiple data bursts in the respective buffer in the memory interface;

for a wrapping memory access request, assigning data required for a beginning and an end of the wrapping memory access request to respective sub-buffers of a single respective buffer to be stored concurrently from a single data burst in the respective sub-buffers of the single respective buffer in the memory interface, the storing of the beginning and end data in a single buffer avoiding the need for an additional data burst to obtain the end data, the data required for the end of the wrapping memory access request being cached in the respective sub-buffer until needed for transfer in response to the wrapping memory access request;

recording a value of a pointer indicating a first sub-buffer of the single buffer storing the end data; and

using the pointer to return to the indicated sub-buffer to retrieve the end data.

8. A method as claimed in claim 7, further comprising, when returning data to the respective device from which a memory access request requiring multiple data bursts over the memory bus was received, reading data out from a first part of single buffer, then reading data out from at least one other of said buffers, then reading data out from a second part of the single buffer.

Claims 9 - 10. (Canceled)

11. A method as claimed in claim 7, further comprising determining whether a received read access request is a wrapping request which requires multiple memory bursts, and, if so, performing the step of assigning each of said data bursts to a respective one of said buffers.

12. A method as claimed in claim 7, wherein the memory controller is a SDRAM controller, and said memory interface receives data from a SDRAM memory device over said memory bus in SDRAM bursts.

13. A programmable logic device, wherein the programmable logic device includes a memory controller, comprising:

at least one bus interface, each bus interface being for connection to at least one respective device formed within the programmable logic device for receiving memory access requests;

a memory interface, for connection to an external memory device over a memory bus;

a plurality of buffers in the memory interface, each of the plurality of buffers sized to store a data burst for a memory access request, each of the plurality of buffers further including a plurality of sub-buffers each sized to store a data beat of the data burst stored in the corresponding buffer; and

control logic, for placing received memory access requests into a queue of memory access requests,

wherein, in response to a received memory access request requiring multiple data bursts over the memory bus, each of said multiple data bursts is assigned by the control logic to a respective buffer of the plurality of buffers in the memory interface, and data from each of said multiple data bursts is stored by the memory interface in the respective buffer, and

wherein, for a wrapping memory access request requiring multiple buffers, data required for each of a beginning and an end of the wrapping memory access request are assigned to respective sub-buffers of a single respective buffer by the control logic, the beginning and end data for the memory access request being stored concurrently from a single data burst in the respective sub-buffers by the memory interface, the storing of the beginning and end data in a single buffer avoiding the need for an additional data burst to obtain the end data, the data required for the end of the wrapping memory request being cached in the respective sub-buffer until needed for transfer in response to the wrapping memory access request; and

wherein the control logic records a value of a pointer indicating a first sub-buffer of the single buffer storing the end data, such that the control logic is able to return to the indicated sub-buffer to retrieve the end data from the single buffer.

Claims 14-17. (Canceled)

10. EVIDENCE APPENDIX

NONE.

11. RELATED PROCEEDINGS APPENDIX

NONE.

61700436 v4